Statistical Techniques in Robotics (16-831, S20) Lecture #1 (Wednesday, February 3)

PWEA, Greedy / Consistent Algorithm

Lecturer: Kris Kitani *Scribes: Manikandtan Chiral Kunjunni Kartha, Erica Weng*

1 Recap

In the previous lecture we discussed about the different kinds of robot learning problems and how the differ. Based on the properties of feedback a robot receives from the environment, we can classify the learning algorithms as following [3]:

1. Exhaustive (robot is exposed to all possible variations of the data) vs Sampled (exposed to only a subset of the variations)
2. Evaluative (the robot receives no feedback about possible alternative actions) vs Instructive (the robot receives the score for all possible actions)
3. Sequence (whether the outcome of a current action affects the next action) vs One-shot (outcome of current action doesn't affect the next action)

Most robot learning techniques like Reinforcement learning suffer from generally weak feedback (evaluative and sequential) as compared to other machine learning techniques like supervised learning and is hence generally harder to solve.

To understand these these properties, we looked at a few common learning techniques and where they lie in the spectrum of feedback properties(See Figure 1).

- Reinforcement Learning: say in the case of autonomous driving, the system initiates actions which influence later states (sequential) and the system does not receive feedback on each possible action it could have taken at a given time step(evaluative) as well as sampled.
- Tabular Reinforcement Learning : say in a *Gridworld* problem, while the current actions influence later states(sequential), the system does receive feedback on each possible action for a state(instructive) and can be exhaustive for small enough problems. The presence of strong instructive and exhaustive feedback helps in designing really efficient algorithms like *Dijkstra's* to solve such problems.

- Multi Armed Bandit: in a bandit system where the reward distribution in not dynamic(rigged), the problem is one-shot, exhaustive and evaluative.

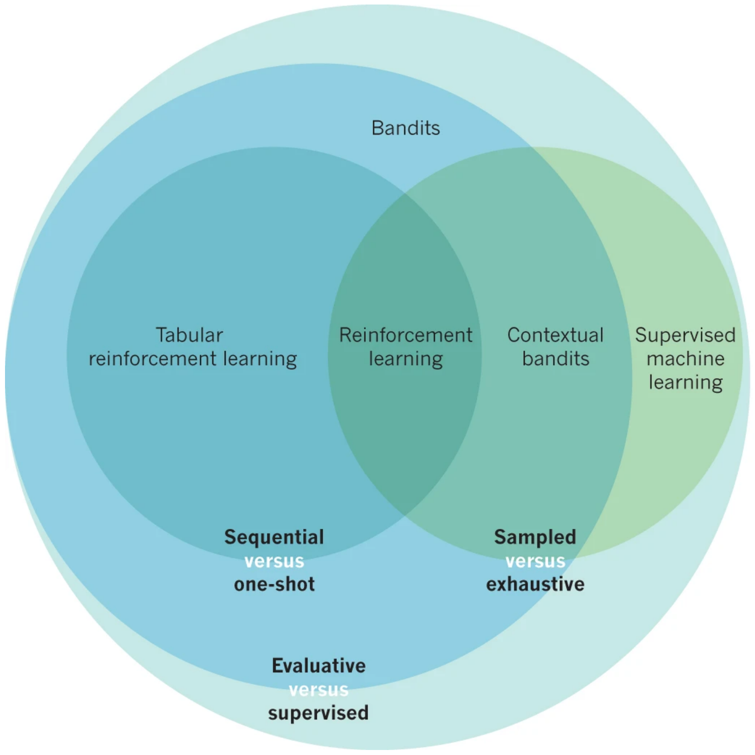
Bandits
Tabular reinforcement learning
Reinforcement learning
Contextual bandits
Supervised machine learning
Sequential versus one-shot
Sampled versus exhaustive
Evaluative versus supervised

Figure 1: Three kinds of feedback in machine learning and examples of learning problems that result from their combination. [3]

2 Online Learning vs. Supervised Learning

The core idea behind online learning is the fact that algorithms makes predictions and learns with each sample input as it is received. In its simplest form the input data is assumed to be one-shot (there is no correlation between samples). However, the feedback maybe instructive or evaluative and exhaustive or sampled. The goal of the online learner is to minimize the loss (when system is realizable – the learner has access to the perfect hypothesis) or more generally – to minimize the regret.

Supervised Learning has clearly separate train and test stages – and follows the assumption that the data observed during training is representative of the data which will be observed during test stage. Online Learning does away this separation and performs incremental learning over each received input. This helps the system adapt in situations where the data distribution received is stochastic or adversarial, something which supervised learning would have trouble with. Online learning technique may also be a better strategy when the training dataset is too large to either fit in memory or be processed at the same time. Interestingly the supervised learning method of Stochastic Gradient Descent (SGD) when trained with a batch size of one may be seen as an online learning technique during its training phase.

3 Prediction With Expert Advice (PWEA)

PWEA is a method of online learning where the system takes decisions based on advice from a set of experts and over time with the goal to be as good as possible as the best expert in the group, if not better. Over time the system learns from making mistakes based on the advice from each expert and updates the hypothesis function which modifies the weighting given to each expert in the group. The different way to choose your hypothesis function and weight update strategy gives us different variations of PWEA.[1, 2]

3.1 Realizability

Definition 1. **Realizability** is the assumption that the learner has access to the perfect hypothesis.

Realizability is an assumption made about the hypothesis space that the learner can achieve at least one perfect hypothesis which will give the global optima for the loss function. Under the conditions of realizability, the performance is measured in terms of the mistake bound, which represents, in retrospect, the cost for not following the perfect hypothesis. In a realistic scenario this may not always be possible and in later methods we would relax this realizability assumption and allow even the best expert to make a few mistakes. In a system of imperfect experts(non-realizability), we will evaluate the performance of the algorithm in terms of the *regret bound* instead of the *mistake bound*. In such a case, we would like to design algorithms which can minimize regrets.

4 Greedy / Consistent Algorithm

With the Greedy/Consistent algorithm, the system uses a learning rule where it picks the first expert in its pool of experts (Greedy) who has been consistently correct in its predictions up until now, and bases its decision solely based on that expert. With each mistake made, the pool of consistent experts reduces by at least 1. Assuming a realizable system where there is at least one perfect expert, the system will be as good as the consistent expert.

There are multiple issues with this strategy, the most obvious one being if realizability is not met, the algorithm will no longer have a pool of experts to pick from and will terminate. The fact that algorithm requires strict consistency also may eliminate an expert who is reasonably good but not perfect if the expert makes a mistake at an early timestep.

Now we define our problem space formally: we have the instance domain $\mathcal{X}$, or the set of possible observations that can be seen any timestep t; the target domain $\mathcal{Y}$, or the set of possible outcomes that can occur; and the hypothesis class $\mathcal{H} = \{h : \mathcal{X} \to \mathcal{Y}\}$, a series of functions mapping an element in the instance domain to an element in the target domain, each of which determines how we will make a prediction based on the observations we see. The Version space $\mathbf{V}^{(t)}$ at time t is defined as the set of experts whom we still trust at time t; all experts in the version space at time t have not yet made a mistake. Then, the Greedy Algorithm pseudocode is listed in algorithm 1.

Algorithm 1 Greedy / Consistent Algorithm

```
1: $\mathbf{V^{(1)}} = \mathcal{H}$                                  ▷ Version space initially stores all hypotheses
2: for $t = 1, \cdots, T$ do
3:     RECEIVE $(\mathbf{x}^{(t)})$                                 ▷ Receive expert prediction
4:     $h$ = FIRSTELEM$(\mathbf{V}^{(t)})$                           ▷ Greedy selection
5:     $\hat{y}^{(t)} = h(\mathbf{x}^{(t)})$                        ▷ predict using first expert in remaining list of experts
6:     RECEIVE $(y^{(t)})$                                          ▷ Receive true outcome
7:     $\mathbf{V}^{(t+1)} \leftarrow \{h \in \mathbf{V}^{(t)} : h(\mathbf{x})^{(t)} = y^{(t)}\}$   ▷ update version list of consistent hypotheses
8: end for
```

4.1 Bounds on the number of mistakes the greedy algorithm will make

Theorem 2. *(Mistake bound of Consistent Algorithm) Let $M_C^{(t)}(\mathcal{H})$ be the total number of mistakes made by the Consistent Algorithm by timestep t for hypothesis class $\mathcal{H}$. Assuming realizability and that there is at least one expert in the starting Version space $^{(1)}$, then $M_C^{(T)}(\mathcal{H})$ is upper-bounded as:*

$$M_C(\mathcal{H}) \leq |\mathcal{H}| - 1$$

Proof. Initially, there are $|\mathcal{H}|$ experts, so:

$$|\mathbf{V}^{(1)}| = |\mathcal{H}|$$

If the consistent algorithm makes a mistake at time t, then:

$$|\mathbf{V}^{(t+1)}| \leq |\mathbf{V}^{(t)}| - 1$$

holds because we remove at least one expert from the Version space $\mathbf{V}$, including the expert whose advice we had just followed. So after M_C mistakes, at step t:

$$|\mathbf{V}^{(t)}| \leq |\mathcal{H}| - M_C$$

This is an upper bound on the size of the version space at time t. For the lower bound: since we assume realizability, then there is at least one expert at any time t:

$$|\mathbf{V}^{(t)}| \geq 1$$

Putting the upper and lower bound together, we get:

$$1 \leq |\mathbf{V}^{(t)}| \leq |\mathcal{H}| - M_C$$

Thus:

$$M_C \leq |\mathcal{H}| - 1$$

□

The implications of theorem 2 are that in the best-case, it will take us at least 1 mistake to discover the true expert (and thus the optimal expert-following strategy). In this best case, all experts except for the perfect expert will fail on the first timestep. In the worst-case, only the expert we decide to trust fails at each mistake made, and thus we must make $M_C \leq |\mathcal{H}| - 1$ mistakes before discovering the true expert.

In terms of time: in the best case, on the first timestep, all experts except for the perfect expert fail, and thus we discover the true expert. In the worst case, the perfect expert and one or more near-perfect experts all do not fail for a large number of timesteps, before the near-perfect experts finally fail. In this case, it takes us many timesteps to discover the true expert.

5 Conclusion

The Consistent algorithm is the simplest of the predicting with expert advice algorithms. Its major flaws are that it requires the strong assumption of realizability, and uses relatively little domain information about the problem. In later lectures we will look at algorithms which slacken the assumptions and do better at incorporating knowledge into the algorithm.

References

[1] N. Cesa-Bianchi, Y. Freund, D. Haussler, D. P. Helmbold, R. E. Schapire, and M. K. Warmuth. How to use expert advice. *J. ACM*, 44(3):427–485, May 1997.

[2] N. Cesa-Bianchi and G. Lugosi. On prediction of individual sequences. *The Annals of Statistics*, 27(6):1865–1895, 1999.

[3] M. L. Littman. Reinforcement learning improves behaviour from evaluative feedback. *Nature*, 521(7553):445–451, 2015.